Gestion des activités pour les enseignants



Leonar Dupuis – FIN2

ETML - Vennes

88 heures

M. Gaël Sonney

(Nom et adresse du mandant)

Table des matières

[1 Spécifications 4](#_Toc167802930)

[1.1 Titre 4](#_Toc167802931)

[1.2 Description 4](#_Toc167802932)

[1.3 Matériel et logiciels à disposition 4](#_Toc167802933)

[1.4 Prérequis 4](#_Toc167802934)

[1.5 Cahier des charges 5](#_Toc167802935)

[1.5.1 Objectifs et portée du projet (objectifs SMART) 5](#_Toc167802936)

[1.5.2 Caractéristiques des utilisateurs et impacts 5](#_Toc167802937)

[1.5.3 Fonctionnalités requises (du point de vue de l’utilisateur) 5](#_Toc167802938)

[1.5.4 Contraintes 5](#_Toc167802939)

[1.5.5 Travail à réaliser par l'apprenti 5](#_Toc167802940)

[1.5.6 Si le temps le permet … 5](#_Toc167802941)

[1.5.7 Méthodes de validation des solutions 5](#_Toc167802942)

[1.6 Les points suivants seront évalués 5](#_Toc167802943)

[1.7 Validation et conditions de réussite 6](#_Toc167802944)

[2 Planification Initiale 7](#_Toc167802945)

[3 Analyse 16](#_Toc167802946)

[3.1 Méthodologie de travail 16](#_Toc167802947)

[3.2 Document d’analyse et conception 18](#_Toc167802948)

[3.2.1 Logo du site 18](#_Toc167802949)

[3.2.2 La maquette 19](#_Toc167802950)

[3.2.3 Schémas de base de données 28](#_Toc167802951)

[3.3 Environnement de projet : limites du système / interfaces avec le monde extérieur 29](#_Toc167802952)

[3.4 Conception des tests 30](#_Toc167802953)

[3.4.1 Test unitaire 30](#_Toc167802954)

[3.4.2 Test d’intégration 31](#_Toc167802955)

[3.4.3 Test de sécurité 31](#_Toc167802956)

[3.5 Planification détaillée 32](#_Toc167802957)

[4 Réalisation 32](#_Toc167802958)

[4.1 Dossier de Réalisation 32](#_Toc167802959)

[4.1.1 Environnement de travail 32](#_Toc167802960)

[4.1.2 Création et connexion à la base de données 34](#_Toc167802961)

[4.1.3 Gestion de l’authentification 36](#_Toc167802962)

[4.1.4 Détails du compte 41](#_Toc167802963)

[4.1.5 Opérations CRUD 45](#_Toc167802964)

[4.2 Modifications 54](#_Toc167802965)

[5 Tests 55](#_Toc167802966)

[5.1 Dossier des tests 55](#_Toc167802967)

[6 Conclusion 57](#_Toc167802968)

[6.1 Bilan des fonctionnalités demandées 57](#_Toc167802969)

[6.2 Bilan de la planification 57](#_Toc167802970)

[6.3 Bilan personnel 57](#_Toc167802971)

[7 Divers 57](#_Toc167802972)

[7.1 Journal de travail 57](#_Toc167802973)

[7.2 Bibliographie 57](#_Toc167802974)

[7.3 Webographie 57](#_Toc167802975)

[8 Annexes 57](#_Toc167802976)

[8.1 Résumé du rapport du TPI / version succincte de la doc. 57](#_Toc167802977)

[8.2 Glossaire 57](#_Toc167802978)

# Spécifications

## Titre

Gestion des activités pour les enseignants

## Description

À l'heure actuelle, l'incorporation d'outils numériques dans le domaine éducatif est devenue cruciale pour améliorer la gestion des diverses activités au sein des établissements scolaires.

Ce projet, qui est un travail pratique individuel, vise à développer une application web pour gérer les activités au sein d'une école, impliquant à la fois des élèves et des enseignants. Cette application permet aux enseignants d'organiser des activités et aux élèves de s'y inscrire. L'accès à l'application nécessite une authentification pour tous les utilisateurs.

Chaque activité, organisée par un enseignant, a un nombre maximum de participants, une date et un lieu définis. Les activités peuvent être de nature sportive telles que le tir à l'arc ou le tennis, ou des sorties culturelles. Les enseignants ont la possibilité de créer, modifier ou supprimer des activités, ainsi que de gérer la liste des participants. Ils peuvent également consulter la liste de leurs propres activités organisées. Les étudiants peuvent s'inscrire à des activités, consulter leurs inscriptions et se désinscrire si nécessaire.

Cette application veut donc répondre aux besoins actuels en matière de gestion des activités scolaires en exploitant les avantages des technologies web pour offrir une solution pratique, accessible et efficace.

## Matériel et logiciels à disposition

Pour démarrer ce projet, il est nécessaire d’avoir à disposition :

* Un PC standard de l’ETML (Windows 10)
* Visual Studio code avec un environnement PHP installé
* Docker
* La suite Microsoft Office pour les documents
* Un dépôt Git

## Prérequis

* Des connaissances en programmation PHP ainsi qu’en programmation orienté objet (POO) sont requises pour réaliser ce projet (références des modules à l’ETML : ICT 403, 404, 226, 120, 411, 133, 326).
* Des connaissances en implémentation et modélisation de bases de données relationnelles (références des modules à l’ETML : ICT 104, 105, 153).

## Cahier des charges

### Objectifs et portée du projet (objectifs SMART)

A compléter. Il s’agit d’ébaucher des réponses aux questions de l’acronyme CQQCOQP (Combien, Quoi, Qui, Comment, Où, Quand, Pourquoi)

### Caractéristiques des utilisateurs et impacts

A compléter… Il s’agit d’identifier le(s) profil(s) de(s) utilisateur-trice(s) type, et les conséquences que cela va avoir sur la conception (couleurs, ergonomie, utilisation, etc.)

### Fonctionnalités requises (du point de vue de l’utilisateur)

A compléter par une espèce de mode d’emploi du produit. S’il s’agissait d’une montre, décrire qu’à part l’heure, il y aura la possibilité d’utiliser un chronomètre, un réveil, …

### Contraintes

Sécurité, backups, disponibilité, système utilisé, interfaces avec autres logiciels, etc.

### Travail à réaliser par l'apprenti

Décrire à quoi doit ressembler le travail produit, ce qu’il faudra rendre …

### Si le temps le permet …

Un objectif complémentaire, si le temps le permet, serait de rendre le site web **responsive**. En effet, il faut prendre en compte le fait que chaque utilisateur a une résolution d’écran différente et dans le cas d’un site non responsive, la navigation pourrait être désagréable et, dans le pire des cas, empêcher l’utilisateur d’accéder à certaines informations à cause des dimensions de certains éléments qui en cacheraient d’autres.

### Méthodes de validation des solutions

Comment les tests vont être entrepris, quels tests doivent être entrepris, etc.…

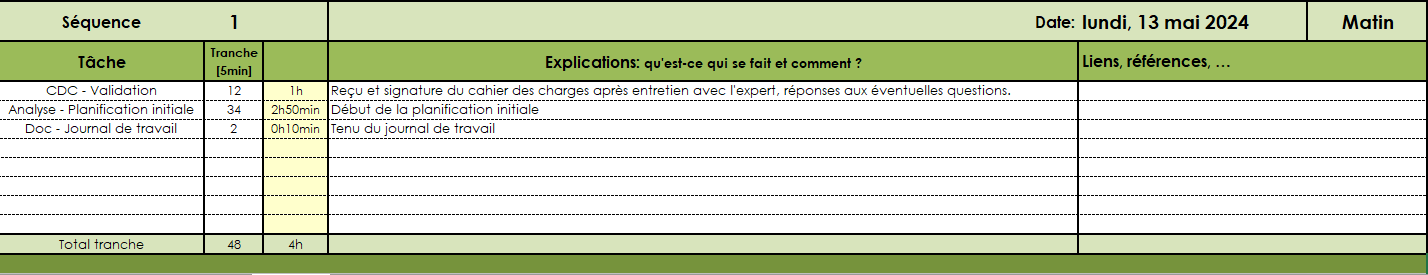
## Les points suivants seront évalués

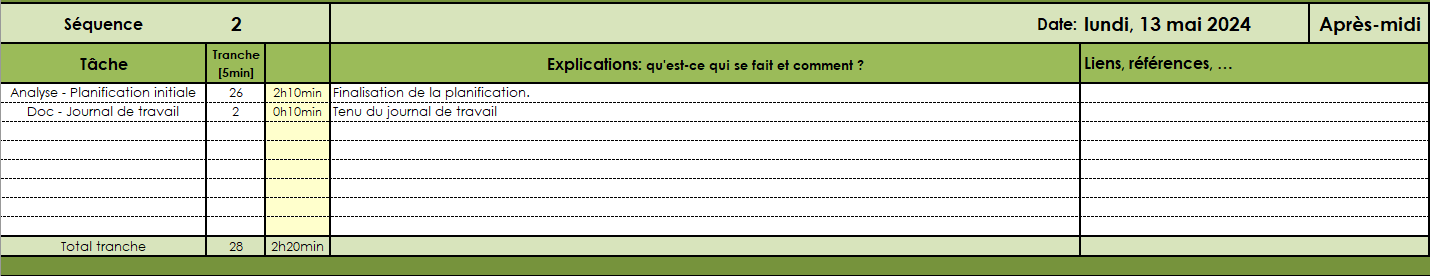
* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Les documentations de mise en œuvre et d’utilisation

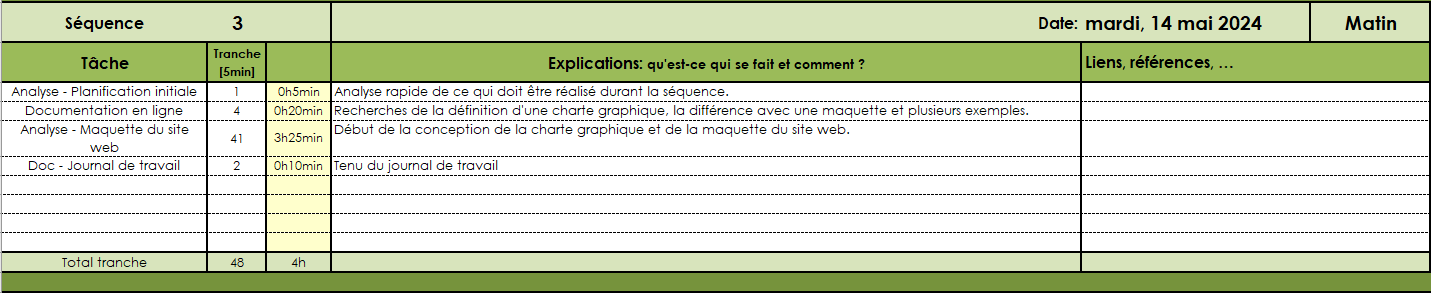
## Validation et conditions de réussite

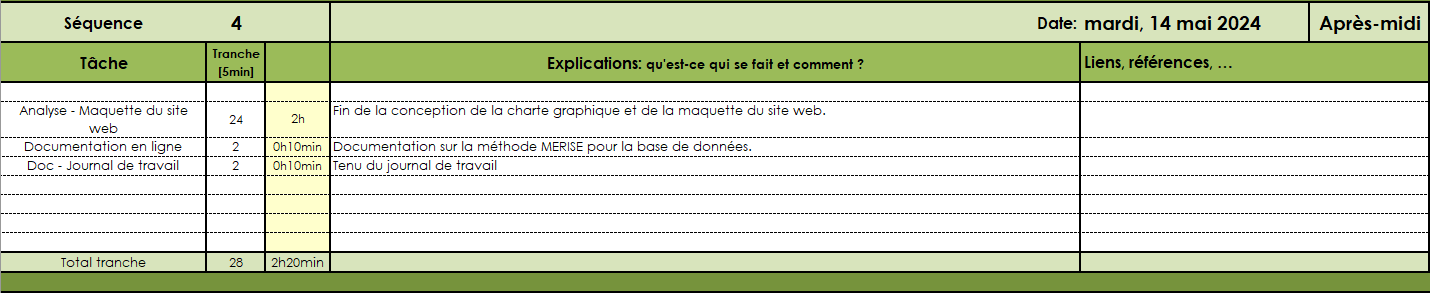
* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

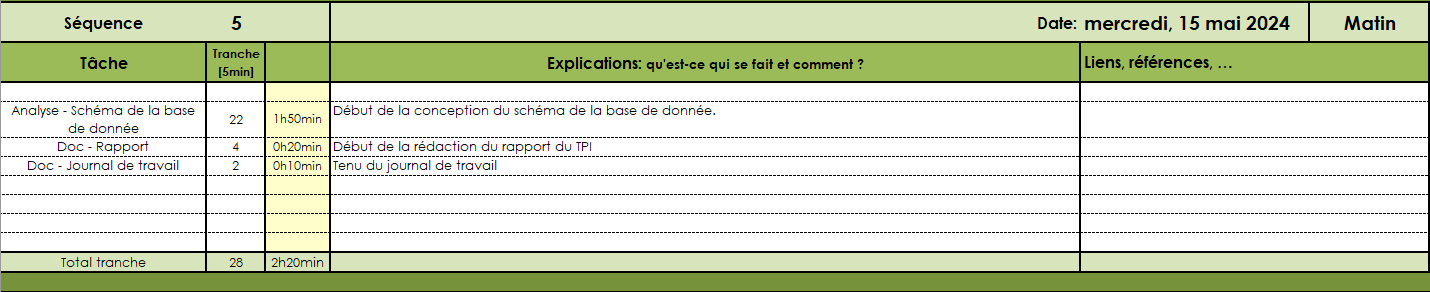
# Planification Initiale

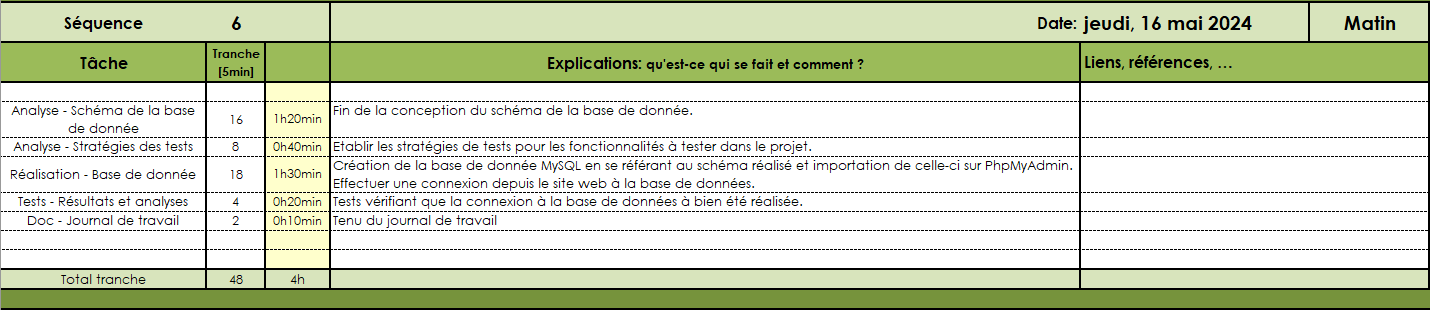


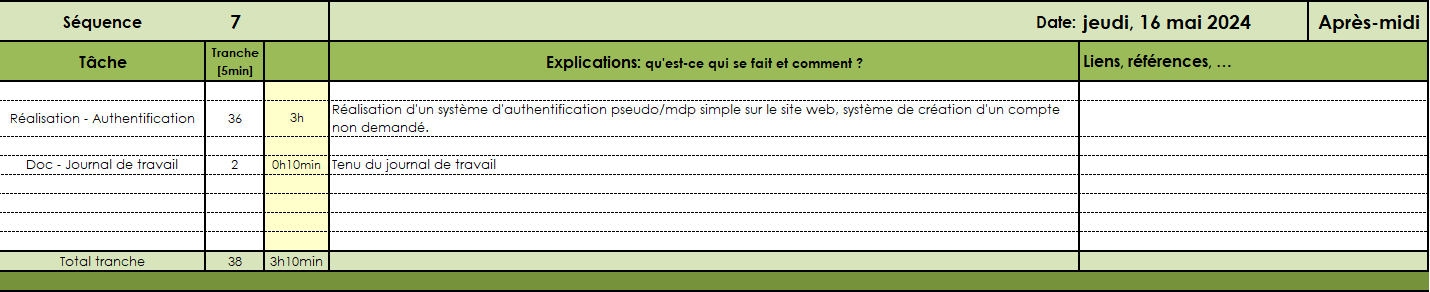


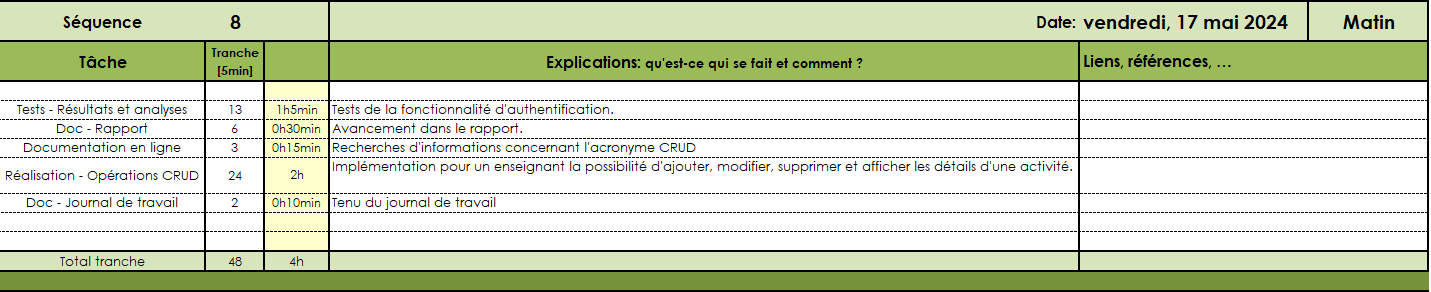


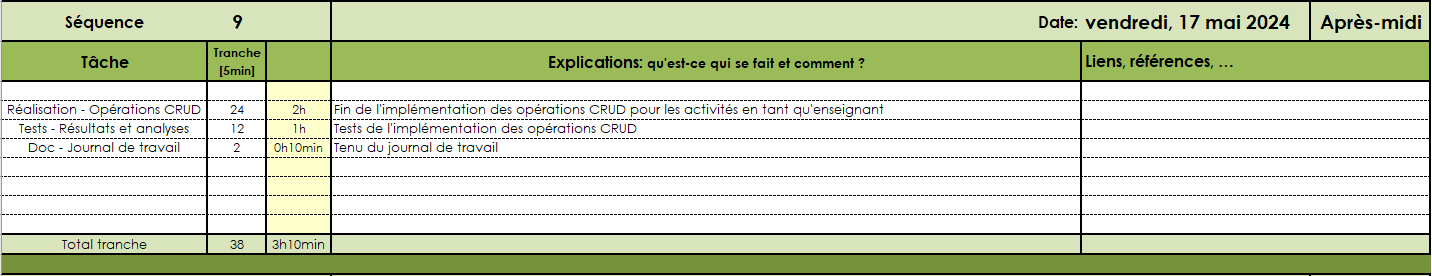


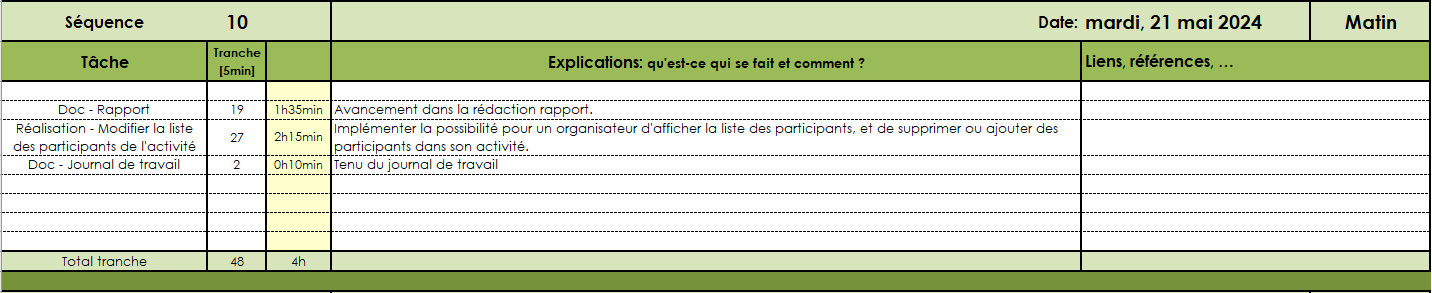


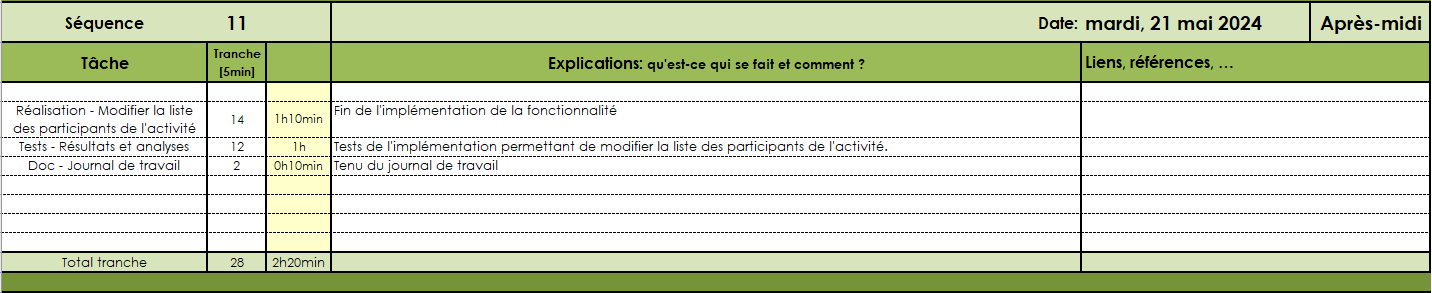


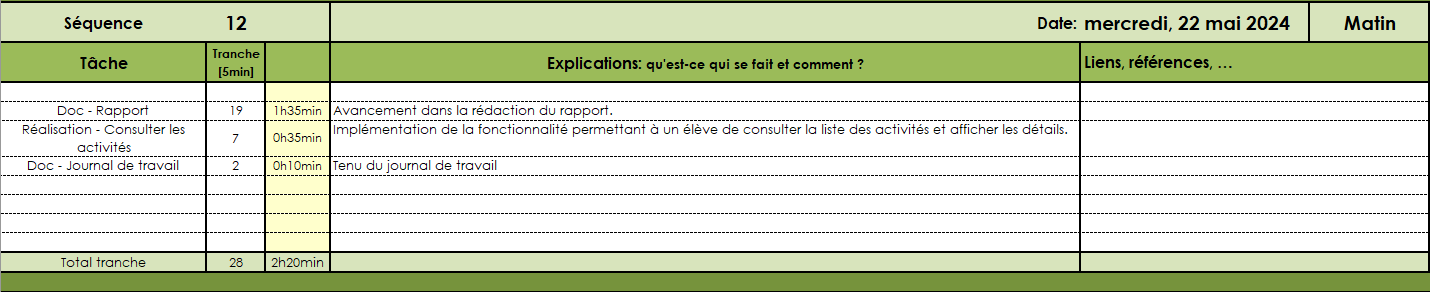


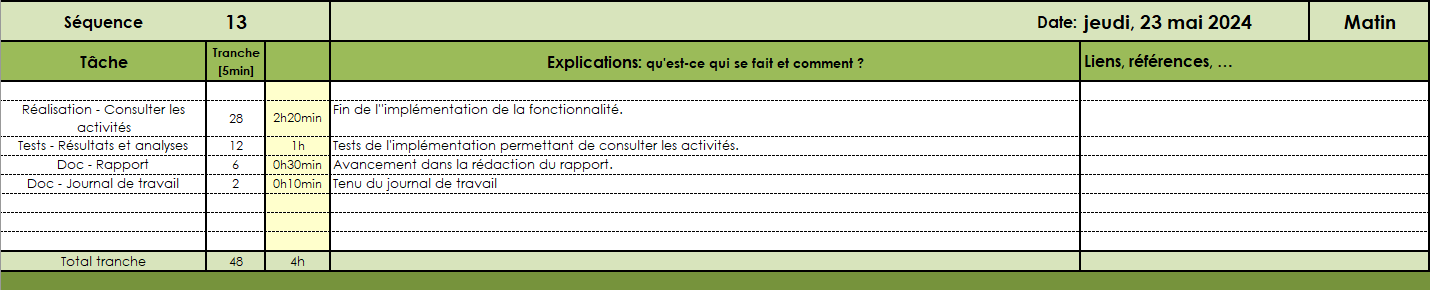


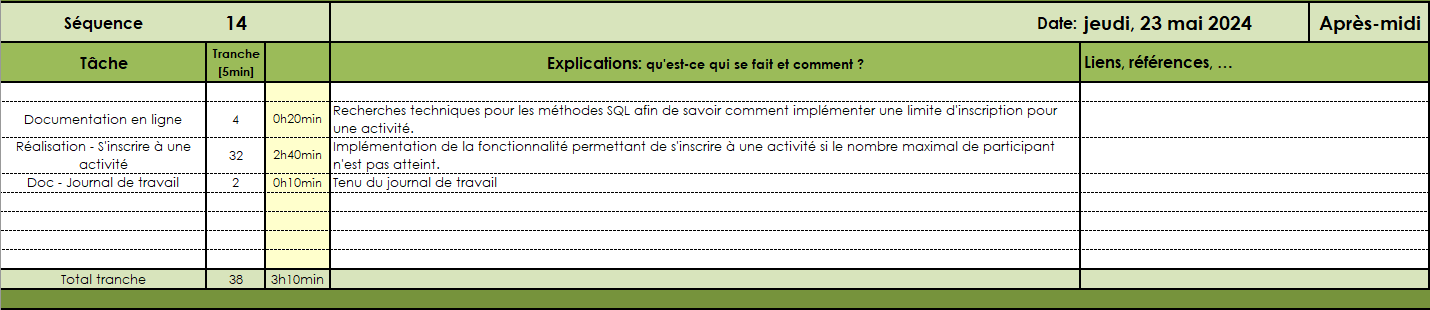


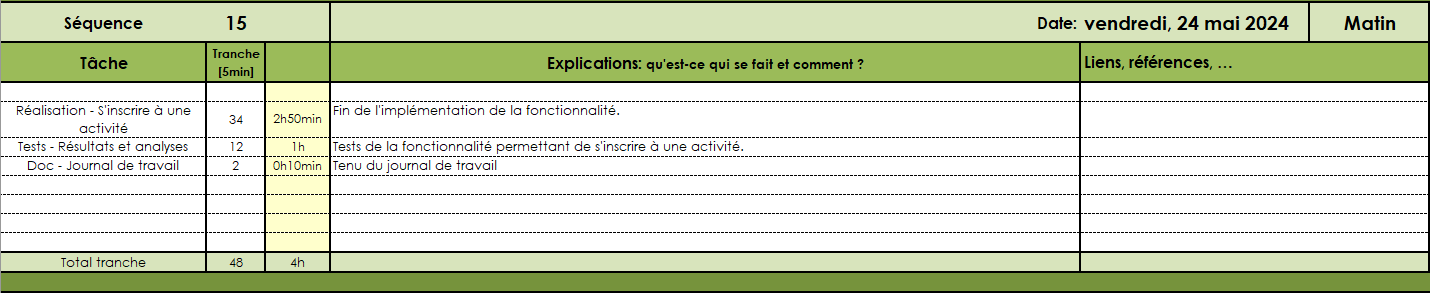


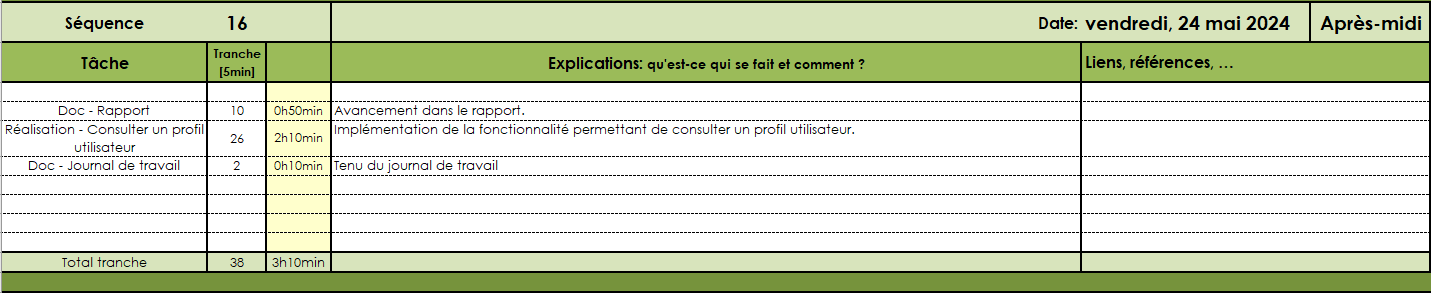


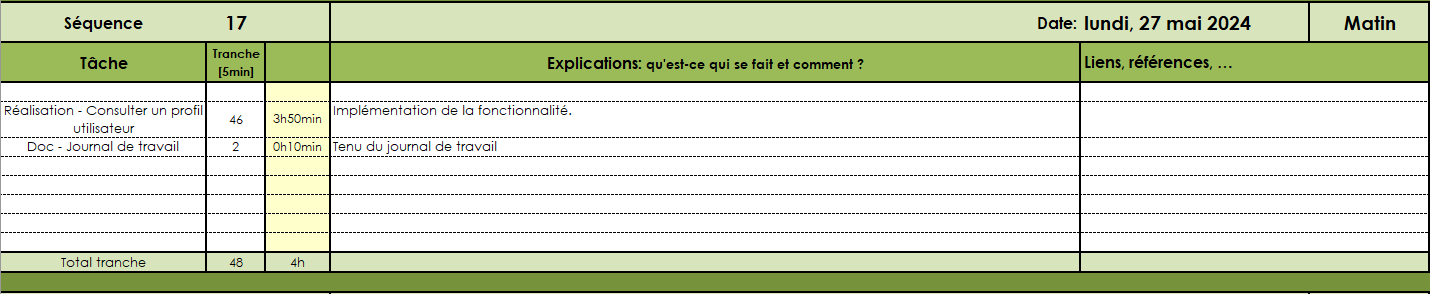


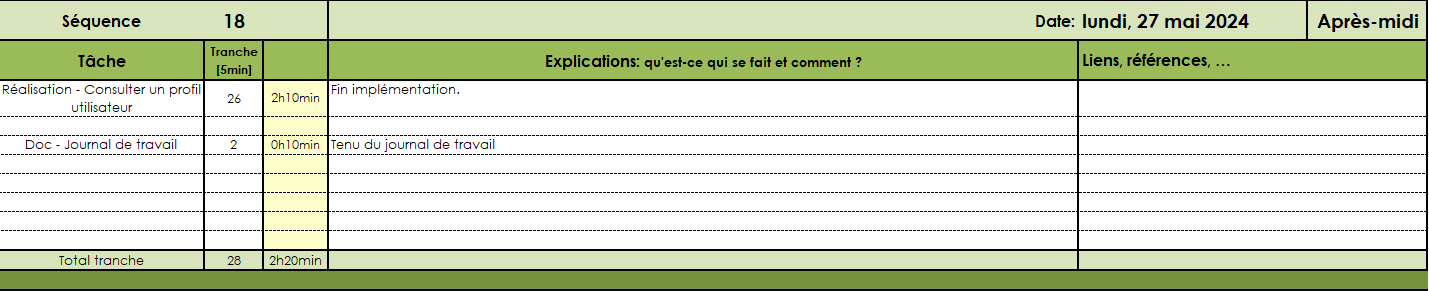


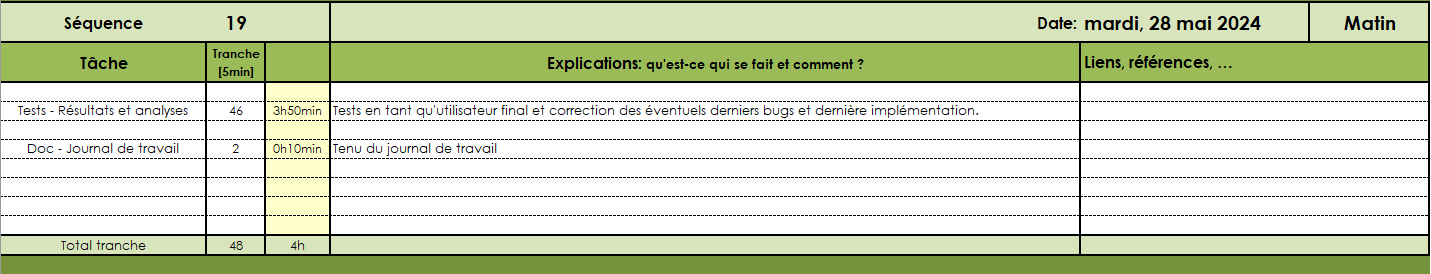


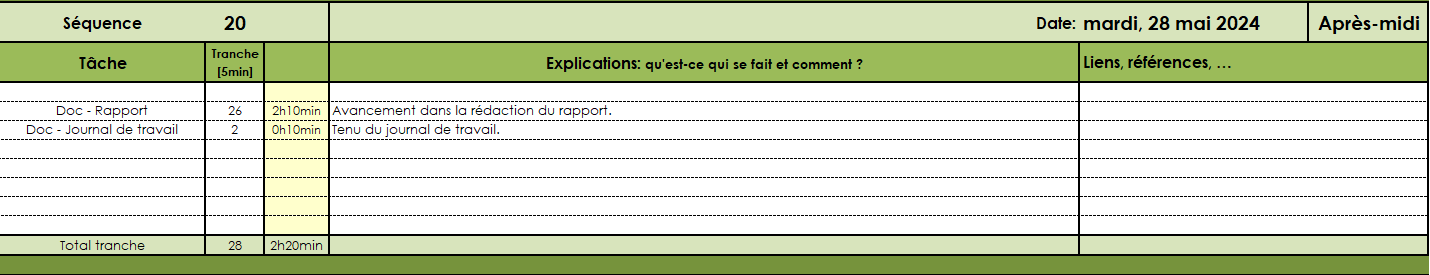


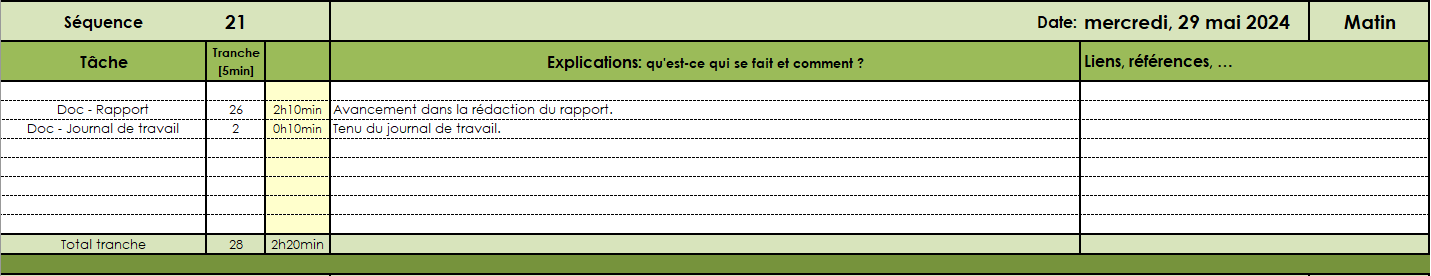


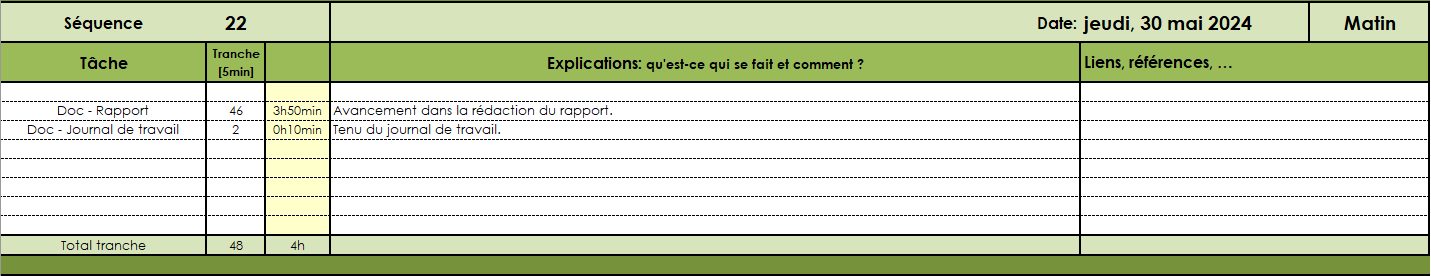


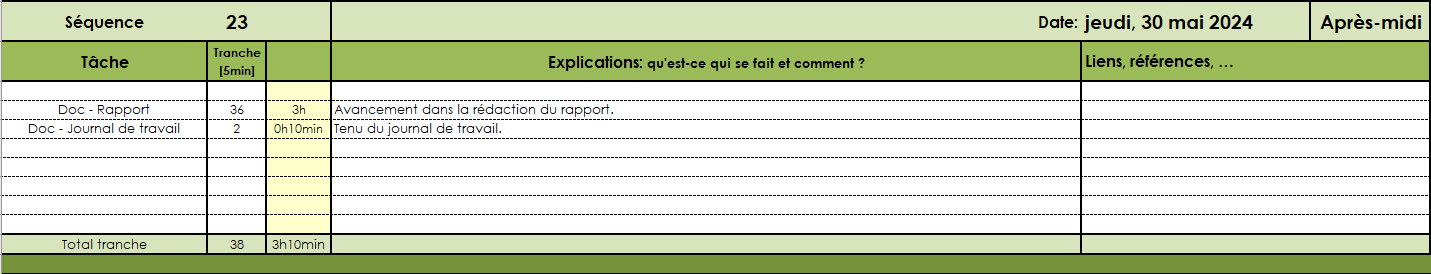


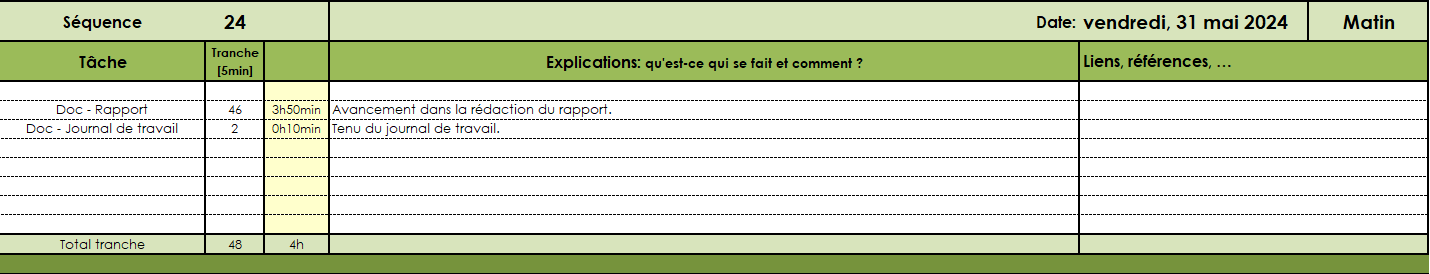


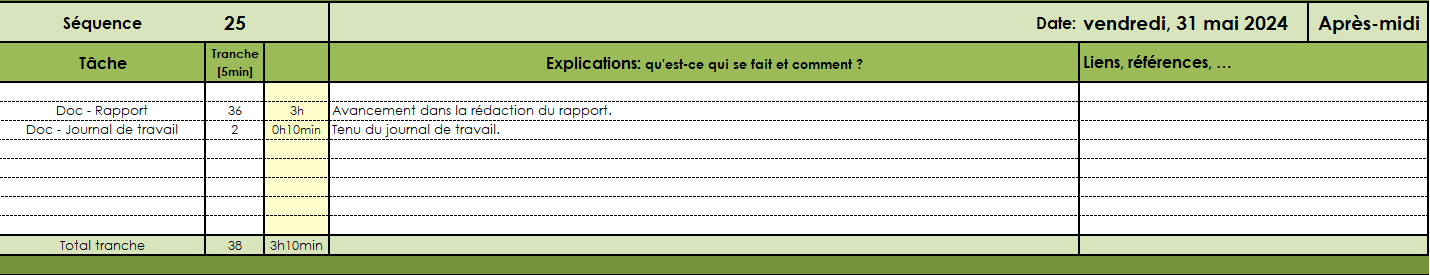


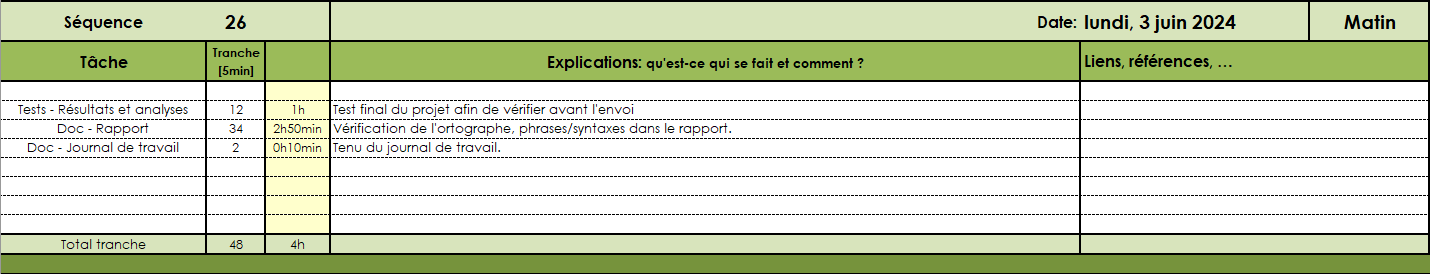


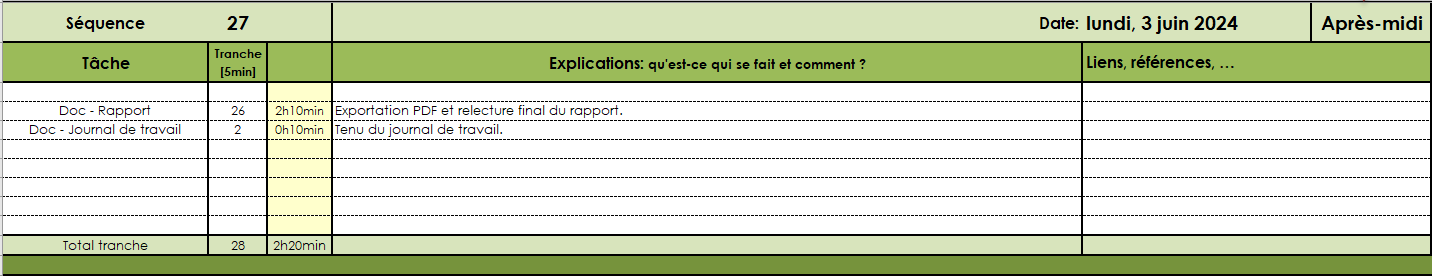












# Analyse

## Méthodologie de travail

Pour ce projet, j'ai délibérément opté pour la méthode Waterfall en raison de sa structure linéaire, offrant ainsi une vision précise et détaillée à chaque étape du développement. Avec des exigences clairement énoncées dans le cahier des charges et un temps de projet relativement court (moins d'un mois), la méthode Waterfall représente pour moi la meilleure approche pour garantir une planification minutieuse et une exécution efficace.

|  |  |
| --- | --- |
| **Etapes** | **Approche** |
| Analyse des besoins | * Les éléments clés du cahier des charges incluent la gestion des activités au sein de l’école, l’inscription des élèves aux activités organisées par les enseignants, et l’authentification des ulilisateurs. * Les fonctionnalités principales sont les opérations CRUD pour les enseignants, ainsi que l’inscription/désinscription pour les élèves. * La contrainte est le nombre maximum de participants pour chaque activité. |
| Conception | * Pour réaliser les maquettes web, j’aurai besoin de Figma. * Les schémas de bases de données seront réalisés à l’aide du logiciel Looping. * Le logo sera réaliser grâce au modèle DALL-E, une extension de ChatGPT 4. |
| Implémentation | * Je vais utiliser des langages de programmation comme HTML, CSS, PHP. * Le code source sera organisé avec un contrôle de version à l’aide de Git pour assurer un bon historique de version et une sauvegarde de données. * Un SSD sera utilisé pour bénéficier d’un backup. * Les modifications et ajouts au code seront gérés à l’aide de push sur la branche principale git. |

|  |  |
| --- | --- |
| **Etapes** | **Approche** |
| Tests | * Les tests incluront des tests unitaires pour chaque fonctionnalité, des tests d'intégration pour vérifier l'interaction entre les composants, et des tests fonctionnels pour valider le comportement global de l'application. * Des tests continus seront faits. Chaque fonctionnalité implémentée sera directement testée. |
| Déploiement | * (A analyser.) |
| Maintenance | * (A analyser.) |

## Document d’analyse et conception

### Logo du site

Tout d’abord, avant de m’attaquer au design de mon site web, j’ai utilisé **ChatGPT** 4 avec le modèle DALL-E intégré pour me générer un logo attrayant, rappelant les activités sportives. Le **prompt** était le suivant :

*« Génère-moi une image de logo d’un site web pour gérer les activités sportives d’un site (organisateur à élèves). Je veux un rendu énergique. »* Le premier résultat était le suivant :



Figure - Premier logo généré

Le résultat m’a plu, l’**IA** a bien intégré l’adjectif « énergique » que je lui avais demandé, mais le fond n’était pas blanc et aucune personne n’était visible sur le logo, ce qui me dérangeait. Alors je lui demandé de me refaire une proposition avec cette fois-ci un fond blanc et une personne, je suis arrivé à ce résultat au bout de la quatrième génération :



Figure - Logo final

Cette génération m’a finalement convaincu. Le fond blanc et l’esthétique colorée du logo me laissent une large palette de possibilités pour la réalisation de la maquette du site, en restant dans le thème d’un « site de gestion d’activités ».

### La maquette

Pour réaliser la maquette de mon site web, j’ai utilisé **Figma** et me suis concentré essentiellement aux pages de fonctionnalités, tout d’abord d’une vue d’utilisateur « élève » puis d’une vue utilisateur « enseignant » qui a des droits supplémentaires comparé aux élèves. M’étant beaucoup inspiré du logo, l’esthétique du site tourne autour de la couleur verte afin d’avoir une bonne cohérence graphique. La maquette de ce site web a été réalisée en me fixant comme objectif de respecter les critères UX : une interface cohérente, crédible, simple, non chargée inutilement et interactive.



Figure - Page d'accueil



Figure - Page de connexion



Figure - Connexion échouée

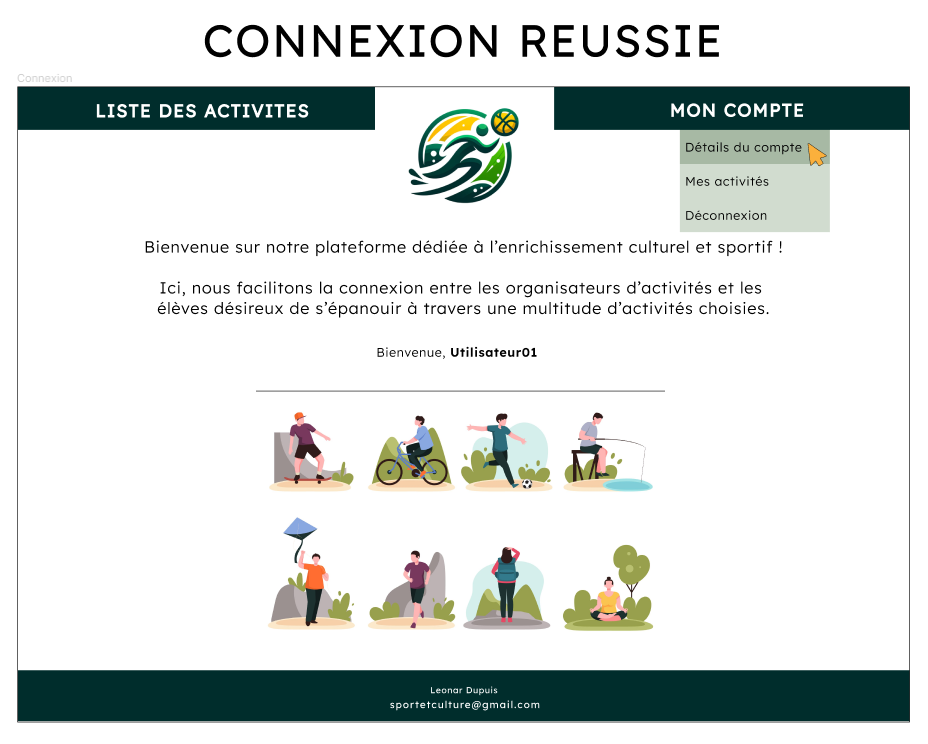


Figure - Connexion réussie



Figure - Détails du compte

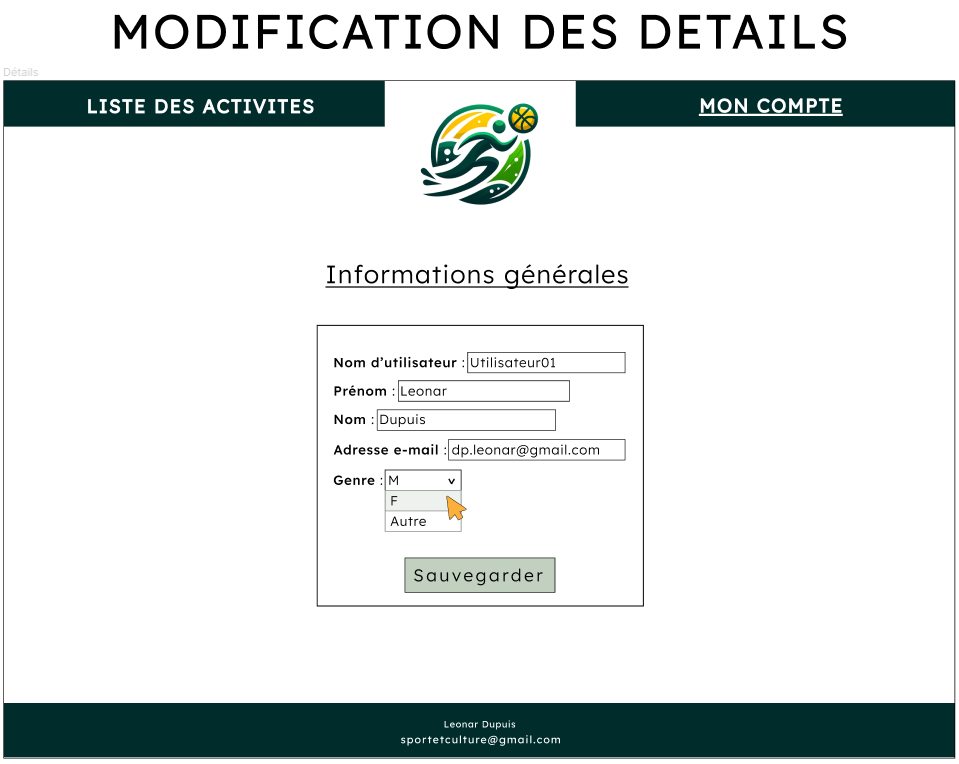


Figure - Modification des détails



Figure - Mes activités (vue élève)



Figure - Mes activités (vue organisateur)



Figure - Modifier une activité (organisateur)



Figure - Créer une activité (organisateur)



Figure - Liste des activités



Figure - inscription à une activité réussie (élève)



Figure - inscription à une activité réussie (élève)



Figure - Détails & Liste participants



Figure - Profil élève



Figure - Profil enseignant

### Schémas de base de données

Pour la modélisation de ma base de données, j’ai utilisé le logiciel Looping qui est très facile à prendre en main. Les schémas suivants ont été réalisés en prenant en compte la convention de nommage standards de l’ETML.

Schéma **MCD** :

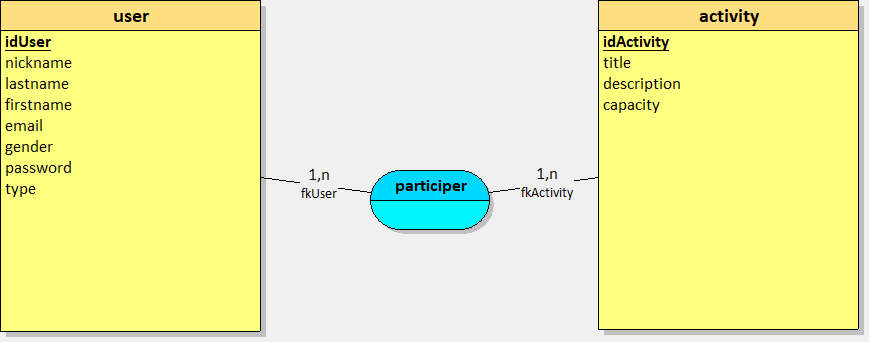


Figure - Schéma MCD

Schéma **MLD** :

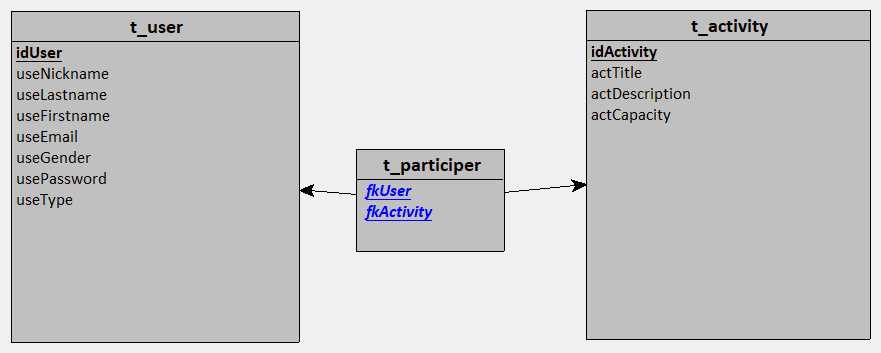


Figure - Schéma MLD

Schéma **MPD** :

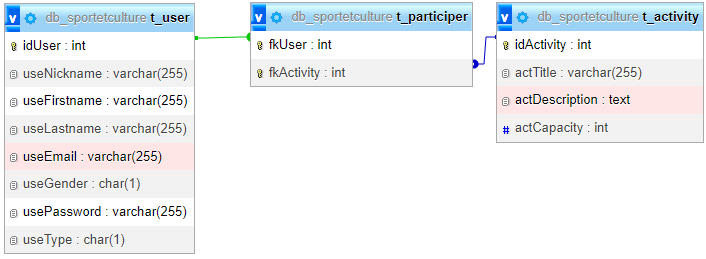


Figure - Schéma MPD

La conceptualisation de ma base de données a été basé sur la méthode MERISE, qui encourage à utiliser les formes normales afin d’assurer l’intégrité et l’efficacité des données. J’ai donc créé mes schémas en m’assurant de respecter les 3 premières formes normales :

* La **première** forme normale veut que chaque table ait une clé primaire unique, (exemple : ‘idUser’). Chaque valeur est atomique, ce qui veut dire que chaque entrée dans une table est une valeur simple et indivisible (comme « useLastname »).
* La **deuxième** veut aucune dépendance partielle. Concrètement, en prenant l’exemple de la table t\_activity, le titre, la description et la capacité de l’activité dépendent entièrement de idActivity.
* La **troisième** forme veut qu’aucun attribut d’une table n’ait de dépendance envers un autre attribut non-clé. C’est-à-dire qu’il n’y a aucune dépendance transitive. Par exemple, dans la table t\_user, le nom, le prénom et l’email de l’utilisateur ne dépendent pas d’autres champs non-clés.

## Environnement de projet : limites du système / interfaces avec le monde extérieur

A Analyser.

## Conception des tests

Les stratégies de test permettent de détecter et de corriger les erreurs afin d'assurer le bon fonctionnement des fonctionnalités et offrir une expérience utilisateur optimale. En utilisant des tests unitaires, d'intégration, fonctionnels et de sécurité, on peut identifier les problèmes avant qu'ils n'affectent les utilisateurs finaux.

Les tests sont planifiés après chaque implémentation de fonctionnalité. Cela permet de maintenir une intégrité globale de l’application, permettant d’éviter les risques de conflits et maintenir une bonne cohésion dans l’ensemble du système.

En fonction du temps disponible, un test final en tant qu’utilisateur final sera inclus afin de garantir une validation complète de l’application.

### Test unitaire

Les tests unitaires visent à vérifier le bon fonctionnement d’éléments précis dans le code, telles que les fonctions et les méthodes, de manière isolée.

La démarche de mes tests unitaires est la suivante :

* **Sélection des fonctions**

J’identifie les fonctions à tester dans mon code.

* **Cas de test**

Pour chaque fonction à tester, je définis les entrées que je vais faire et les résultats attendus.

* **Exécution manuelle**

J’exécute manuellement ces fonctions avec les différentes entrées et je compare les résultats obtenus avec les résultats attendus. Si le travail obtenu n’est pas satisfaisant, je revisionne mon code avant de réeffectuer la procédure de test.

### Test d’intégration

Les tests d’intégration vérifient que les différents modules du code fonctionnent bien ensemble.

La démarche de mes tests d’intégrations est la suivante :

* **Scénarios d’intégration**

J’identifie une fonctionnalité qui interagit avec différents modules de mon site web.

* **Cas de test**

Je définis un scénario de test, où je suis par exemple un utilisateur qui va provoquer l’interaction de plusieurs modules entre eux en utilisant une fonctionnalité. Je définis plusieurs scénarios de tests, cherchant plusieurs chemins d’exécution possibles.

* **Exécution manuelle**

J’exécute ce scénario en comparant les résultats obtenus avec les résultats attendus.

### Test de sécurité

Les tests de sécurité permettent d’identifier les vulnérabilités potentielles dans une application web. C’est un point très sensible, surtout quand notre site contient du PHP.

La démarche de mes tests de sécurité est la suivante :

* **Vulnérabilités connues**

J’identifie les types courants de vulnérabilités (ex : injection SQL).

* **Cas de test**

Je définis un ou plusieurs tests pour chaque type de vulnérabilités.

* **Exécution manuelle**

Je tente d’exploiter ces vulnérabilités en utilisant des entrées malveillantes, par exemples dans des formulaires ou des points de saisie.

## Planification détaillée

A ce stade, après l’analyse complète du projet, un planning détaillé et complet (avec tâches, sous-tâches, dépendances, durée, …) peut être finalisé.

Le planning détaillé doit s’inscrire dans le planning initial. Il faut que l’on puisse situer cette planification détaillée par rapport à la planification initiale.

# Réalisation

## Dossier de Réalisation

### Environnement de travail

Tout d’abord, il est impératif d’avoir un bon environnement de travail pour pouvoir commencer ce projet. Le projet étant versionné sur GitHub, un **readme** expliquant comment l’environnement a été configuré se trouve à la racine de l’arborescence du site.

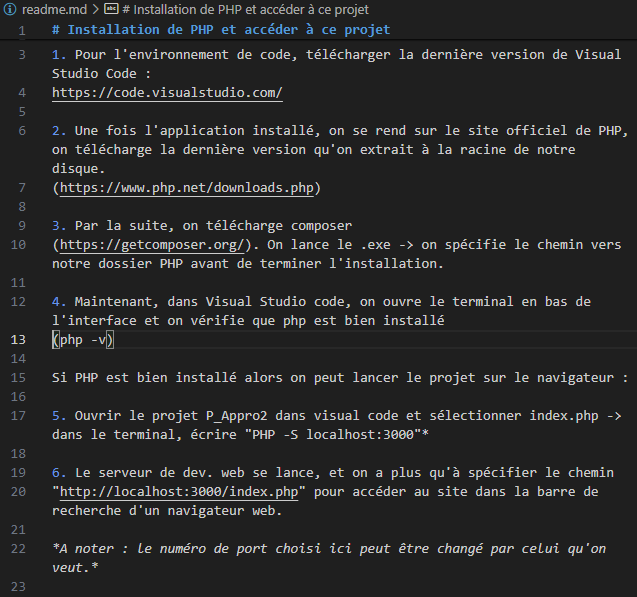


Figure - readme p.1

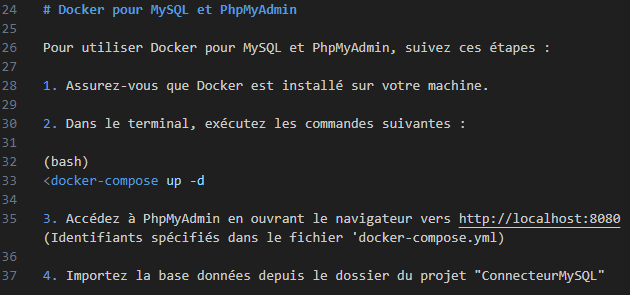


Figure - readme p.2

En ce qui concerne l’organisation des fichiers en tant que tel pour le projet, une structure **MVC** a été mis en place selon la logique suivante :

* Les fichiers qui sont montrés/vus par des utilisateurs sont placés dans le répertoire « views ».
* Les fichiers gérant des conditions, des erreurs avant d’être envoyés dans une méthode vont passer dans le répertoire « controllers ».
* Les fichiers qui interagissent avec la base de données via des fonctions, sont dans le répertoire « models ».

|  |  |
| --- | --- |
| Figure - models / controllers | Figure - views |

### Création et connexion à la base de données

Le début de ce projet a consisté pour moi à créer, dans mon dossier ConnecteurMySQL, ma base de données « db\_sportetculture » puis de créer mes tables en me basant sur le modèle créé avec l’application Looping :

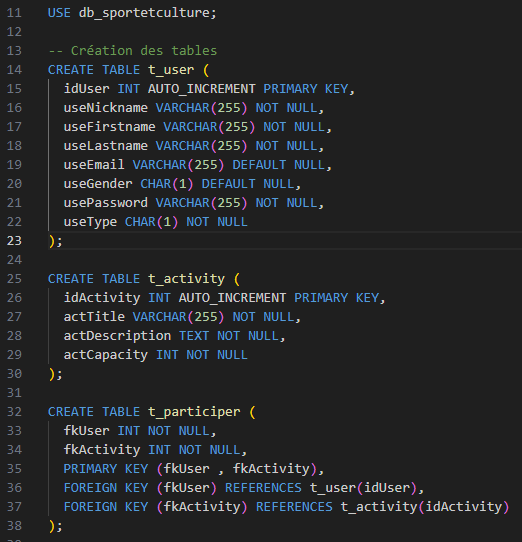


Figure - Création des tables SQL

Maintenant que la base de données est prête, on se rend sur PhpMyAdmin en utilisant les champs par défaut : **root**, **root.** Puison crée la base de données, en laissant par défaut le choix de caractères :

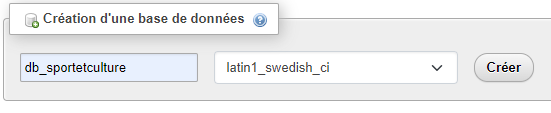


Figure - Création de la base de données

On importe ensuite notre code de création des tables, et on vérifie que l’importation ait bien réussie :

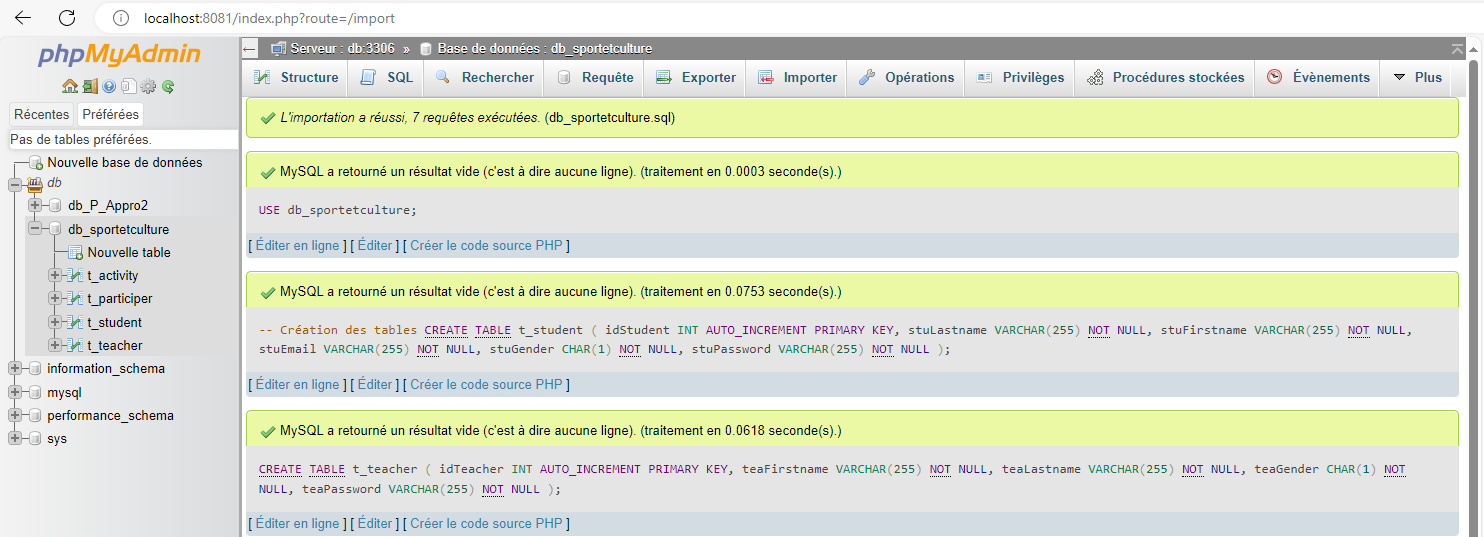


Figure - Importation réussie

Une fois importée, on établit la connexion entre notre site web et la base de données. Dans la partie « models » de mon arborescence web, là où les données sont manipulées, on crée le fichier database.php.

Dans notre classe Database, dans la fonction \_\_construct(), on se connecte via PDO (extension définissant l'interface pour se connecter à une base de données avec PHP) en spécifiant les informations à rentrer sur mon PhpMyAdmin comme le mot de passe. Si la connexion est établie, un message «db connecté» apparaît sur le site, sinon, un message d’erreur survient.

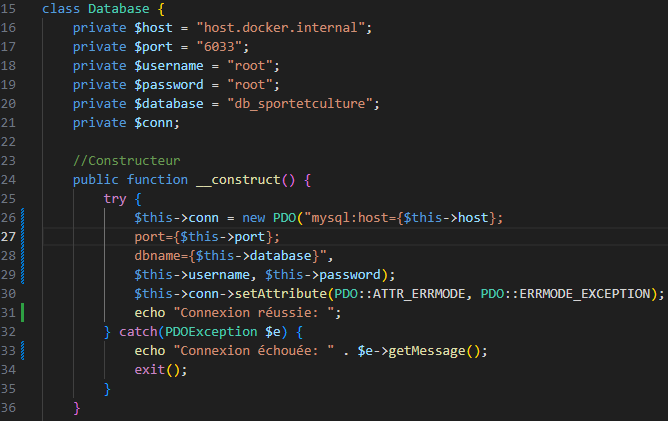


Figure - Connexion à la base de données

La connexion ici marche donc sans problème, grâce au message visible à l’accueil :



Figure - Connexion réussie

### Gestion de l’authentification

Sur le site web, un système de connexion a été mis en place avec la possibilité de s’inscrire bien que cette fonctionnalité soit optionnelle. Elle est disponible pour l’utilisateur en haut à droite dans ma barre de navigation.



Figure - Barre de navigation (visiteur)

Arrivé sur la page, l’utilisateur doit remplir un formulaire basique avec un mot de passe à confirmer.

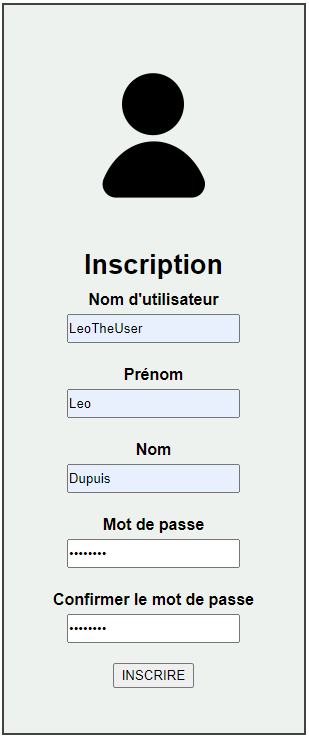


Figure - Inscription

Cette page est présente dans le répertoire « views » dans mon site web, c’est-à-dire qu’il s’agit de la partie visible pour l’utilisateur, mais la page de vérification des champs (gestion d’erreurs), elle, va être dans ma partie controllers et va vérifier que l’utilisateur n’a pas sauter un champ :



Figure - Vérification des champs

Si l’utilisateur a bien rempli tous les champs, alors le formulaire va passer dans la partie « models », dans une fonction « registerUser » qui contient comme paramètres les champs du formulaire à enregistrer dans la base de données.

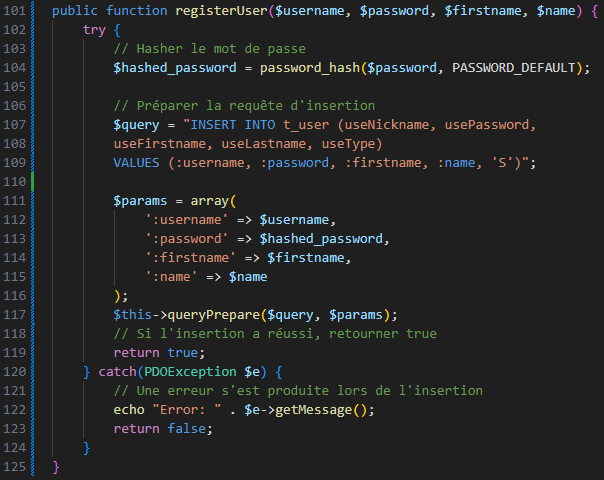


Figure - Fonction "RegisterUser"

Le mot de passe sera **haché** afin de sécuriser les informations confidentielles de l’utilisateur.



Figure - Mot de passe haché

Dans la méthode précédente, on peut voir qu’une méthode « queryPrepare » est appelée. Celle-ci se composte en trois étapes :

* Préparer une requête SQL
* L’exécuter avec des paramètres
* Gérer les erreurs.

C’est une bonne pratique qui aide à prévenir les erreurs, ou encore les injections SQL.

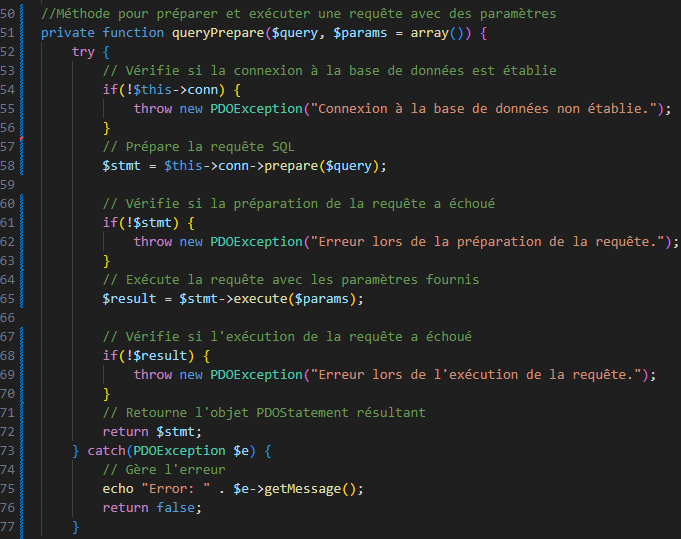


Figure - QueryPrepare

Pour revenir à l’utilisateur, on peut voir que celui-ci a bien été enregistré dans la base de données :

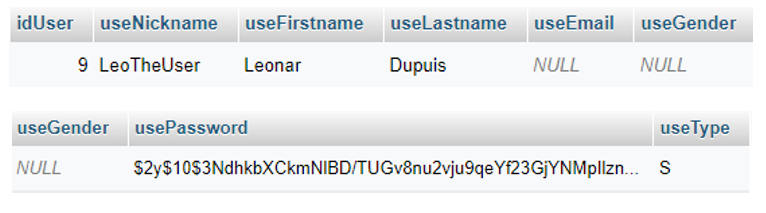


Figure - Utilisateur inscrit

L’utilisateur peut maintenant se connecter.



Figure - Connexion

Malgré tout, si les informations ne correspondent pas avec le bon nom d’utilisateur et son mot de passe, un message survient.

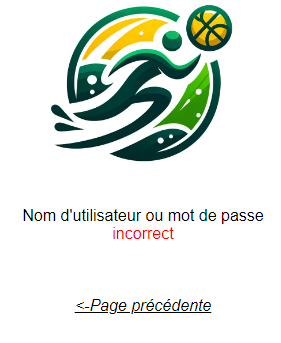


Figure - Vérification connexion

La méthode utilisée ici vérifie juste que le nom d’utilisateur est le mot de passe sont corrects. Elle créée une requête qui, une fois préparée et exécutée, va être utilisée pour comparer le mot de passe fourni avec le mot de passe stocké (après le déchiffrement). Si le résultat est correct, les informations seront stockées dans la session, sinon, retourne false.

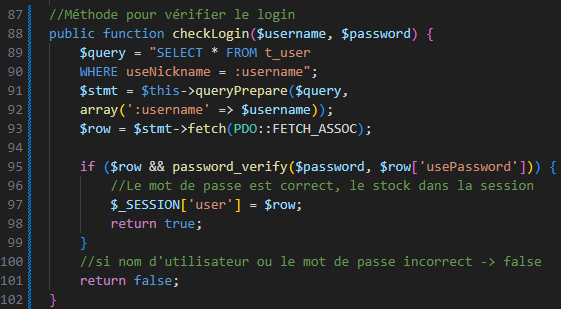


Figure - Fonction checkLogin

### Détails du compte

L’utilisateur a la possibilité, s’il le souhaite, modifier des informations sur son compte s’il le souhaite : un nouvel onglet est disponible grâce à la connexion.

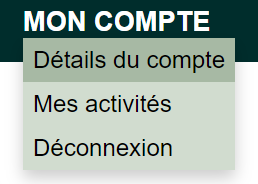


Figure - onglet "Détails du compte"

Arrivé sur la page « userDetails.php », toutes les informations, même le statut « d’Etudiant », est spécifié et la possibilité d’étoffer son profil (ajouter son e-mail ou son genre) est possible. Si l’utilisateur était un enseignant, le titre « Enseignant » indiqué à la place.



Figure - Informations générales

Une fois appuyé sur le bouton « modifié », les champs deviennent modifiables.

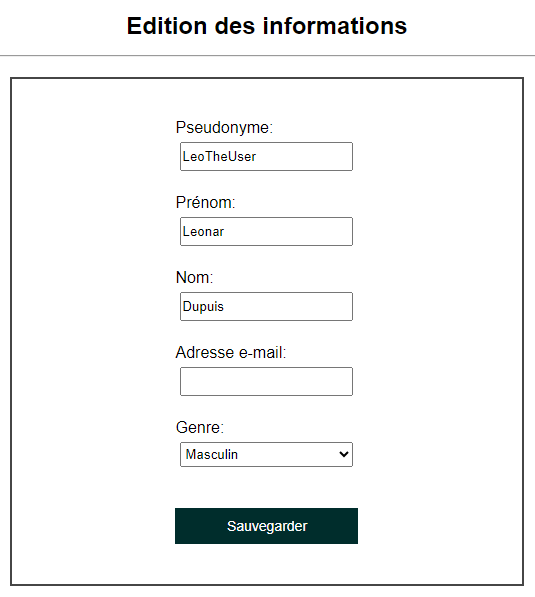


Figure - Modification infos utilisateur

Arrivé sur la page « userDetails.php », toutes les informations, même le statut « d’Etudiant », est spécifié et la possibilité d’étoffer son profil (ajouter son e-mail ou son genre) est possible.

La saisie des champs restent tout de même stricte : Si l’utilisateur insère des chiffres dans son prénom, par exemple, un message d’erreur survient :

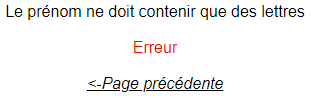


Figure - Erreur champ utilisateur

Pour mettre à jour les données de l’utilisateur, la page « userDetailsCheck.php » dans mon controllers s’occupe d’appeler la méthode updateUserInfo.

A partir de l’identifiant de l’utilisateur, on utilise la requête UPDATE pour effectuer une mise à jour avec les nouvelles informations entrées par l’utilisateur précédemment. Si la méthode retourne ‘true’, la mise à jour a réussie, sinon un message d’erreur est montré à l’utilisateur.

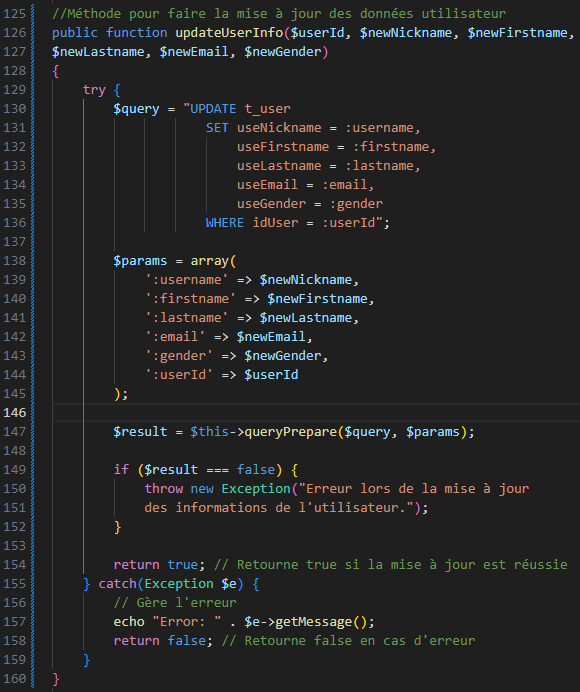


Figure - méthode updateUserInfo

### Opérations CRUD

Les opérations de base « CRUD » concernent ici les interactions entre les activités et les utilisateurs, et plus précisément les enseignants.

En prenant un compte utilisateur enseignant, on peut tout d’abord créer une activité en accédant à l’onglet « Mes activités » en haut à droite de la page (myActivities.php).

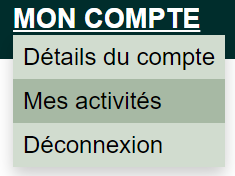


Figure - onglet "Mes activités"

Un message apparaît ici, que ce soit en tant qu’élève ou enseignant, que l’utilisateur est affilié à aucune activité. L’enseignant ici a le droit de créer une activité :

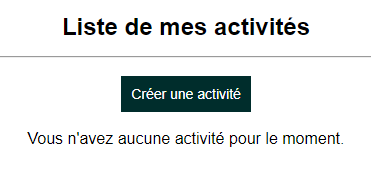


Figure - Liste de mes activités (vide)

En cliquant sur le bouton, un formulaire permettant de créer l’activité apparaît.

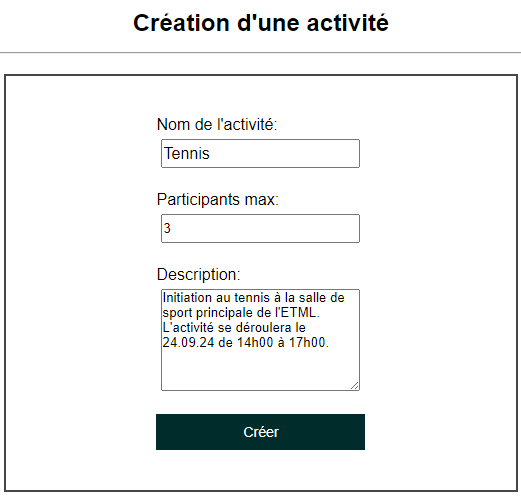


Figure - Création d'une activité

Si l’enseignant oublie d’indiquer une information, un message lui dira de remplir tous les champs. Dans un autre cas, le nombre de participants est négatif, un message survient

|  |  |
| --- | --- |
| Figure - erreur champs vides | Figure - Erreur champ négatif |

Une fois le bouton « créé » appuyé à la fin du formulaire, la méthode createActivity dans la partie « models » va insérer une nouvelle activité dans la table t\_activity avec les informations fournies : titre, description et capacité comme vu dans le formulaire précédent.

L’association est également effectuée entre le créateur de l’activité (l’enseignant) et cette activité dans la table t\_participer.

La méthode retourne l’ID de l’activité créée si réussie, sinon la méthode retourne false.

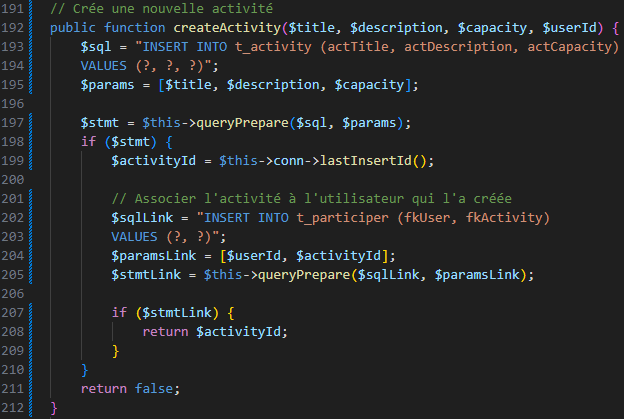


Figure - méthode createActivity

Une fois de retour sur sa page personnelle, l’utilisateur peut voir la nouvelle activité avec plusieurs actions disponibles (un message lui indique si l’activité a bien été créée ou non).



Figure - Activité créée

On vérifie dans la base de données que l’activité a bien été créée. Pour ce faire, on consulte les nouveaux champs dans les colonnes des tables *t\_activity et t\_participer.*

|  |  |
| --- | --- |
| Figure - t\_activity | Figure - t\_participer |

Ce tableau est rempli grâce à cette méthode qui récupère les activités pour un utilisateur selon son type (enseignant ou élève). Si c’est un élève :

* Le nom de l’organisateur de l’activité est envoyé à l’utilisateur (*organizerName*)

Si c’est un enseignant :

* La méthode récupère juste les informations des activités.

Si la méthode échoue, la méthode retourne un tableau vide.



Figure – Méthode getActivitiesForUser

Dans le cas où l’on souhaite apporter des modifications à notre activité pour une quelconque raison, un bouton « modifier » est disponible pour l’enseignant. S’il clique dessus, un formulaire prérempli avec les informations précédentes sont affichées, avec la possibilité bien sûr de modifier les champs.



Figure - Modification d'une activité

Les conditions pour l’admission de ce formulaire sont les mêmes que pour la création, mais avec une subtilité cette-fois ci.

Si l’enseignant souhaite baisser le nombre de participants en dessous du nombre inscris, cela ne va pas être possible et un message va s’afficher.

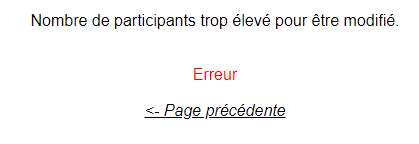


Figure - Erreur nombre de participants élevé

Pour gérer cette erreur, on appelle la méthode **getParticipantCount**.

Celle-ci compte le nombre d’élèves participant à une activité.

La requête utilise la fonction COUNT(\*) et va chercher le nombre d’utilisateurs de type ‘S’ (useType = ‘S’), donc élève, pour une activité donnée.

En effet, l’enseignant étant aussi un participant, je ne veux pas qu’il soit comptabilisé dedans. En prenant le cas de figure où une activité possède une capacité de 5 places, si on ajoute l’enseignant, alors celle-ci en possède en réalité 4.

La fonction retourne finalement le nombre de participants, et si aucun résultat n’est trouvé : 0.

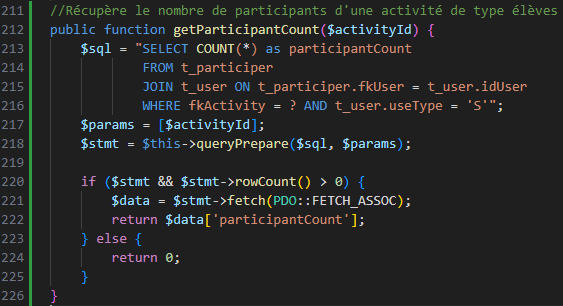


Figure - méthode getParticipantCount

Finalement, la capacité de l’activité sera comparée avec le nombre de participants. Si elle est plus petite, alors la modification est impossible.



Figure - Vérification nombre de participants

Si toutes les conditions sont réunies pour envoyer le formulaire, alors on appelle la méthode **updateActivity** pour appliquer les modifications dans la base de données.

Celle-ci permet la mise à jour des informations d’une activité (titre, description, capacité) sélectionnée par l’enseignant en mettant à jour les colonnes ‘*actTitle’*, ‘*actDescription’* et ‘*actCapacity’* de la table *t\_activity*. L’activité est reconnue grâce à $activityId.

Le résultat est ‘true’ si la mise à jour a réussi, sinon ‘false’.

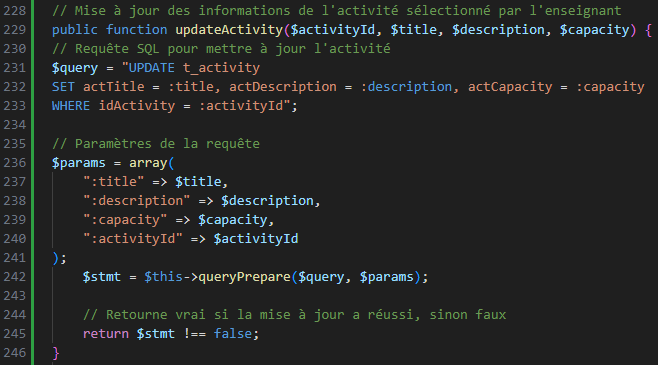


Figure - méthode updateActivity

L’utilisateur sera ensuite redirigé sur la page de ses activités, avec un message lui signalant si l’activité a bien été modifié ou non.

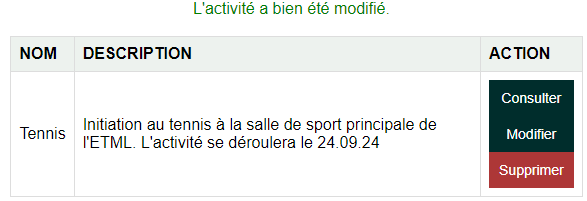


Figure - Activité modifiée

Une des 4 opérations CRUD indispensable ici est la suppression d’une activité. Toujours sur le tableau précédent, en cliquant sur le bouton « Supprimer », un pop-up\* apparaît sur l’écran de l’enseignant.

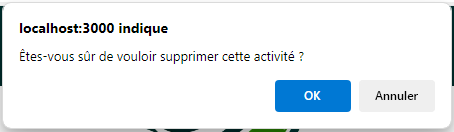


Figure - pop-up supprimer l'activité

Une des 4 opérations CRUD indispensable ici est la suppression d’une activité. Toujours sur le tableau précédent, en cliquant sur le bouton « Supprimer », une boite de dialogue\* apparaît sur l’écran de l’enseignant lui demandant de confirmer la suppression de l’activité.

Du javascript est utilisé ici avec une fonction appelée **confirmDelete** pour supprimer l’activité via une fenêtre pop-up. Lorsque l’utilisateur clique sur ‘confirmer’, l’utilisation du ‘fetch’ permet d’envoyer la requête ‘DELETE ‘ vers la partie controllers (*deleteActivity.php*) avec l’ID de l’activité à supprimer. Si la réponse est correcte (‘*response.ok’*), la page est rechargée pour actualiser l’affichage, sinon il y’a une erreur.

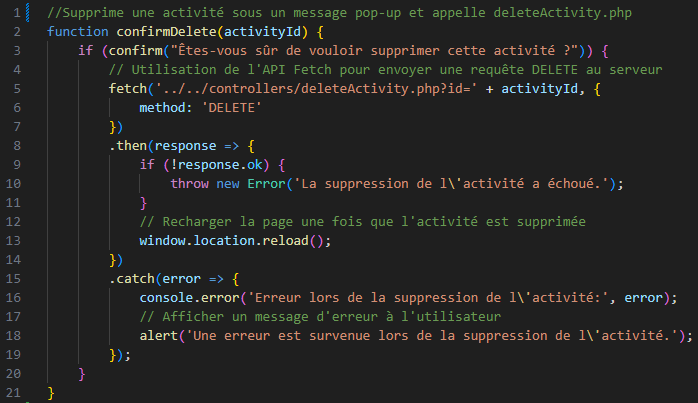


Figure - script.js

Pour revenir sur le côté PHP, ma page *deleteActivity.php* va servir à appeler la méthode **deleteActivity**.

Cette méthode a pour objectif de supprimer une activité spécifique grâce à la requête DELETE qui va supprimer toutes les lignes de la table *‘t\_participer’* où la colonne *‘fkActivity’* correspond à l’identifiant de l’activité. Cela évite tout scénario de bug où un participant est encore associé à une activité qui n’existe plus.

Par la suite, la requête DELETE est utilisée pour supprimer la ligne de la table *‘t\_activity’* où la colonne *‘idActivity’* correspond à l’identifiant de l’activité.

Pour finir, si les deux suppressions ont réussi, la méthode retourne ‘true’. En revanche, si l’une des suppressions a échoué, la méthode retourne ‘false’.

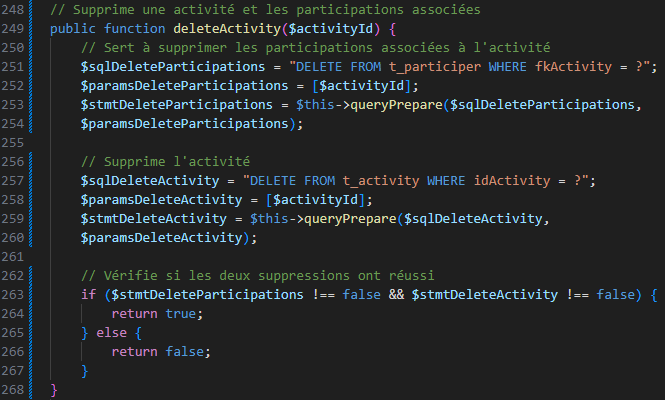


Figure - méthode deleteActivity

|  |  |
| --- | --- |
| Figure - Erreur lors de la suppression | Figure - Activité supprimée |

La dernière opération CRUD intégrée sur le site permet la consultation des détails d’une activité. Pour accéder aux détails d’une activité, il faut juste cliquer sur le bouton « Consulter » à côté de l’activité souhaité, par exemple depuis notre profil « Mes activités ».



Figure - Détails de l'activité

Cette page contient quelques particularités, plus précisément concernant l’affichage de l’organisateur ou encore du statut de l’activité.

La première méthode consiste tout simplement à récupérer les détails d’une activité à partir de son identifiant. Une gestion d’erreur avec un bloc « try-catch » est utilisé ici pour encapsuler le code qui pourrait potentiellement lancer une exception. Le résultat est retourné sous forme de tableau associatif dans $activityDetails, sinon, retourne ‘false’.

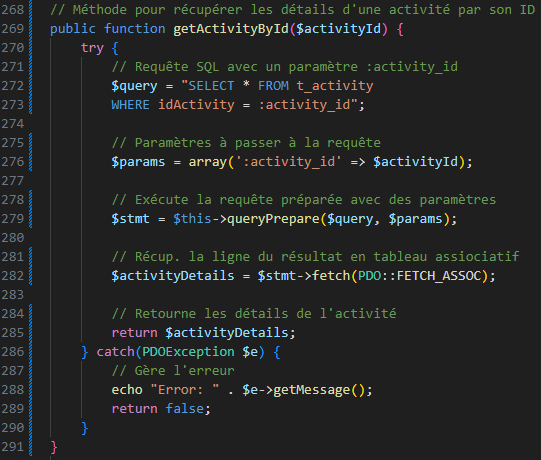


Figure - méthode getActivityById

La deuxième utilise aussi l’identifiant de l’activité comme paramètre, mais elle est plus sélective que la précédente dans la manière où elle ne récupère que les informations des utilisateurs (prénom/nom) de type enseignant associés à l’activité spécifié grâce à une jointure. Le résultat est également retourné sous un tableau associatif, sinon, retourne ‘null’.

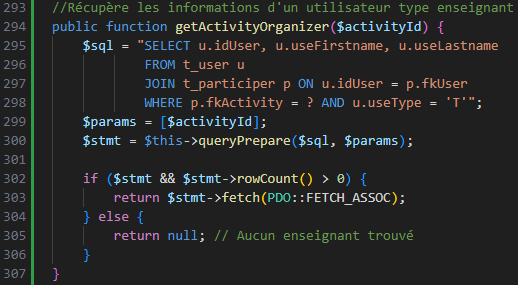


Figure - méthode getActivityOrganizer

La dernière méthode utilisée pour le statut vérifie si la capacité maximale d'une activité est atteinte en comparant le nombre actuel de participants de type élèves (grâce à une jointure) avec la capacité maximale définie de base.

La requête COUNT compte le nombre actuel de participants afin d’effectuer une condition vérifiant que *‘actCapacity’* est supérieur à *‘currentParticipants’*, sinon, on retourne ‘false’.

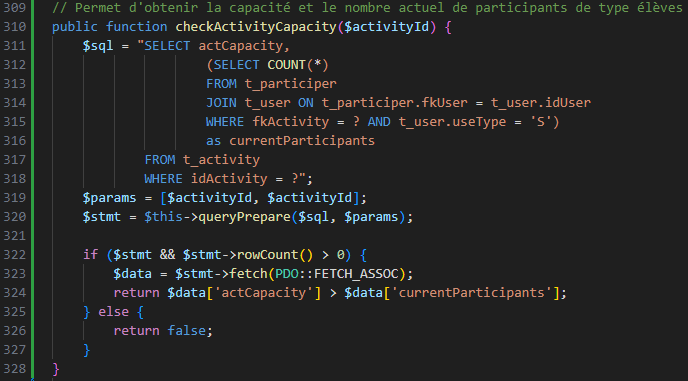


Figure - méthode checkActivityCapacity

L’inscription n’étant pas encore réalisé à ce stade, on crée et on insert les élèves dans une des activités depuis la base de données dans l’objectif de vérifier que le statut de l’activité change. On voit ici que chaque utilisateur est différent, mais associé à la même activité dans la table *t\_participer*. L’enseignant ici n’est pas comptabilisé grâce à la jointure.

|  |  |
| --- | --- |
| Figure - t\_participer - 4 élèves | Figure - activité indisponible |

### Liste des participants d’une activité





Dans la barre de recherche, une fonction va permettre de faire ressortir les utilisateurs correspondants aux lettres recherchées, avec la possibilité de les ajoutés à la liste des activités (inscription forcée par l’enseignant).



Sur le plan technique, cette fonctionnalité requiert

Cette partie permet de reproduire ou reprendre le projet par un tiers.

Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :

Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)

Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)

Code source commenté des éléments logiciels développés.

Modèle physique d’une base de données.

Arborescences des documents produits.

Il faut décrire le parcours de réalisation et justifier les choix.

## Modifications

Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.

Date, raison, description, etc.

# Tests

## Dossier des tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fonctionnalité** | **Procédure de test** | **Résultat attendu** | **Résultat obtenu** | **Correction** |
| **Base de données** |  |  |  |  |
| **Authentification** |  |  |  |  |
| **Opérations CRUD** |  |  |  |  |
| **Modifier la liste des participants** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fonctionnalité** | **Procédure de test** | **Résultat attendu** | **Résultat obtenu** | **Correction** |
| **Consulter les activités** |  |  |  |  |
| **S’inscrire à une activité** |  |  |  |  |
| **Consulter un profil utilisateur** |  |  |  |  |

# Conclusion

## Bilan des fonctionnalités demandées

Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

Si ce n’est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différence entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.

# Divers

## Journal de travail

Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu’une activité de recherches a été entreprise, il convient d’énumérer ce qui a été trouvé, avec les références.

## Bibliographie

Références des livres, revues et publications utilisés durant le projet.

## Webographie

Références des sites Internet consultés durant le projet.

# Annexes

## Résumé du rapport du TPI / version succincte de la doc.

-Situation de départ

-Mise en œuvre

-Résultats

## Glossaire